VaanNila

Menu

Struts 1

Struts 2

Spring

Hibernate

Ant

Log4j

Struts 2 Annotations Example

We will learn annotations in struts 2 using the hello user example. In this example we will get the user name and display a welcome message to the user. There are two versions of this example, in the first one we will see how to do this by using the intelligent defaults provided by the struts 2 framework. We will not do any configuration in this example except the deployment descriptor.

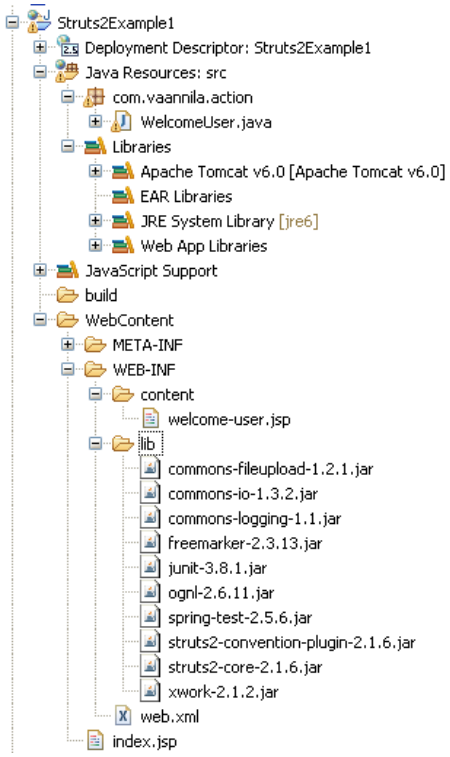
The example is created using ecilpse. The war file of this example is also provided at the end of this tutorial so that you can try it yourself.

So lets start, you need to have the following jar files in the WEB-INF/lib directory.

```
commons-fileupload-1.2.1
commons-io-1.3.2
commons-logging-1.1
freemarker-2.3.13
junit-3.8.1
ognl-2.6.11
spring-test-2.5.6
struts2-convention-plugin-2.1.6
struts2-core-2.1.6
xwork-2.1.2
```

You can definitely save a lot of time by having the correct versions of these jar files in the lib directory. The `struts2-convention-plugin-2.1.6` jar file is needed if your are using annotations.

This is the directory structure of the hello user example.



Now we will create the index page. This page is simple, we use the struts tags to create the page. The `textfield` tag is used to create the textfiled and the `submit` tag is used to create the submit button. The index.jsp page contains the following code.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
           8">
        <title>Hello World</title>
    </head>
    <body>
        <s:form action="welcome-user" >
            <s:textfield name="userName" label="User Name" />
            <s:submit />
        </s:form>
    </body>
</html>
```

Note the URL value of the action attribute in the form tag. In the end we will see how everything relates together.

Subscribe

RSS

Email

We compose the welcome message in the `execute()` method of the WelcomeUser class and we return "success". The WelcomeUser class contains the following code.

```
01. package com.vaannila.action;
02.
03. import com.opensymphony.xwork2.ActionSupport;
04.
05. public class WelcomeUser extends ActionSupport{
06.     private String userName;
07.     private String message;
08.
09.     public String execute() {
10.             message = "Welcome " + userName;
11.             return SUCCESS;
12.     }
13.
14.     public void setUserName(String userName) {
15.             this.userName = userName;
16.     }
17.
18.     public void setMessage(String message) {
19.             this.message = message;
20.     }
21.
22.     public String getUserName() {
23.             return userName;
24.     }
25.
26.     public String getMessage() {
27.             return message;
28.     }
29. }
```

Note the class name, can you find any similarities between the action URL and the class name? if yes got the concept, if no don't worry you will learn what it is in the coming pages.

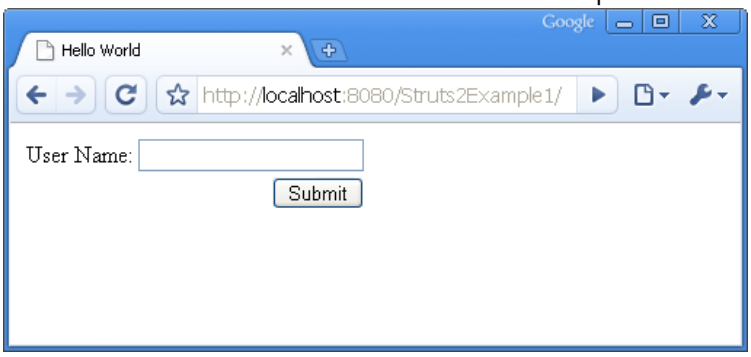
We display the welcome message to the user using the `welcome-user.jsp` page. The content of the page is very simple, we just display the message. The important thing to note here is the name of the page.

```
01. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
02. pageEncoding="ISO-8859-1"%>
03. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
04. <html>
05. <head>
06. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
07. <title>Welcome User</title>
08. </head>
09. <body>
10.
11. <h1>${message}</h1>
12.
13. </body>
14. </html>
```

Now we will configure the web.xml for the struts 2 framework. We need to specify the filter and the filter mapping here. Except this there is no need to have any other XML configuration files.

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
      version="2.5">
03.
04.    <display-name>Struts2Example1</display-name>
05.
06.    <filter>
07.         <filter-name>struts2</filter-name>
08.         <filter-class>
09.             org.apache.struts2.dispatcher.ng.filter.
                  StrutsPrepareAndExecuteFilter
10.         </filter-class>
11.     </filter>
12.
13.     <filter-mapping>
14.         <filter-name>struts2</filter-name>
15.         <url-pattern>/*</url-pattern>
16.     </filter-mapping>
17.
18.     <welcome-file-list>
19.         <welcome-file>index.jsp</welcome-file>
20.     </welcome-file-list>
21.
22. </web-app>
```

Now the coding part is complete. You can run the example by using the following URL "http://localhost:8080/Struts2Example1/"



Enter a user name and submit the form, you will see the following welcome-user.jsp page.

The example works fine. Now lets see how the example works.

The **Convention plug-in** is the one which does everything in the background. The Convention plug-in does the following things.

- By default the Convention plug-in looks for the action classes inside the following packages **strut, struts2, action or actions**. Here our package name is com.vaannila.**action**. Any package that matches these names will be considered as the root package for the Convention plug-in.
- The action class should either implement **com.opensymphony.xwork2.Action** interface or the name of the action class should end with **Action**. Here we extend our WelcomeUser class from **com.opensymphony.xwork2.ActionSupport** which inturn implements com.opensymphony.xwork2.Action.
- The Convention plug-in uses the action class name to map the action URL. Here our action class name is `WelcomeUser` and the URL is `welcome-user`. The plug-in converts the camel case class name to dashes to get the request URL.
- Now the Convention plug-in knows which Action class to call for a particular request. The next step is to find which result to forward based on the return value of the execute method. By default the Convention plug-in will look for result pages in **WEB-INF/content** directory.
- Now the Convention plug-in knows where to look for results, but it doesn't know which file to display inside the content directory. The Convention plug-in finds this with the help of the result code returned by the Action class. If "success" is returned then the Convention plug-in will look for a file name `welcome-user-success.jsp or welcome-user.jsp . It need not be a jsp file it can be even a velocity or freemaker files.` If the result value is "input" it will look for a file name `welcome-user-input.jsp or welcome-user-input.vm or welcome-user-input.ftl`.
- For more Struts 2 annotations example refer here (Struts 2 Annotations - Part 2).

You can download the Struts 2 annotation example by clicking the download link below.

Source :**Download**

Source + Lib :**Download**